Build System

Background

A critical component of Agile Software Engineering is Continuous Integration loosely understood to be integration of individual efforts into the overall system frequently. At the center of such a notion is a build of the entire application whenever an individual engineer releases his/her work. Most teams can ill afford manual builds. There are quite a few build systems such as Jenkins (https://www.jenkins.io), circleci (https://circleci.com) that support such automation.

Tools such as Jenkins are also come with a lot of baggage that hobbyist and otherwise individual engineers may shun. The needs however of ensuring that the code repository is kept updated and work in progress to be integrated and unit tested periodically remains.

In this projectlet we will build the core of a build system. The key features then are:

- Retrieve the source code from a revision control system such as git
- Create the links necessary to trace the build to the source code
- Build the application. Notion of build in this case in need driven - be it to archive or publish build artifacts or run a unit test
- Archive the log of all the activity

Requirement Specs

Let us specify the requirements for a build tool called jobs.

Id	Need
1	Support git based repositories
2	Support distinct branches within git repositories
3	The tool be able to enumerate the branches at the time of initialization or later.
4	Each build shall be assigned a unique identification starting with 1
5	The tool shall support a retention criteria - in terms of the number of builds.
6	The tool shall generate traceability information that can be included in the builds suitable for languages: Ada, C, C++, Python, and Go. The traceability information can also be generated as ini formatted text files.
7	Traceability information required: Branch name Commit id - brief and full Build date and time Build System ie host name where built
8	The tool shall support the following stages of builds: initialization, build, publish, archive. Initialization - retrieve other repositories as necessary Build - build the application Publish - the build artifacts to be published Archive - Capture the build history as an archive
9	The build stages can be driven by a script checked out from the repository
7	

Design Inputs

Req ID	Command Line Example	
1	jobs init [options] git@bitbucket.org:ToprLLC/passwords.git	At the designated workspace, create the top level structure for this job. Clone the default branch
2	—jobspace=<dirname> default: $JOBSPACE	Specify a folder as the jobspace. The repository name is used as the folder for this job's workspace; For the above repository the name will be passwords.
3	—init=scriptdir Optional	Name of a directory in the repository where scripts are to be found. Python3, shell are some of the scripting languages that can be executed.
5	—trace=(C,C++,Ada,Python,Go,Text) default: Ada	Trace information file to be generated in the specified language(s)
6	—retention=number default: 3	
7	jobs build jobname script [branch name]	For the specified branch perform a build as the next build id. If branch name is not specified the default branch name specified during init is used.
		The following environment variables will be set before executing the script: BUILD_NUMBER OUTPUT_DIR SOURCE_DIR REVISION_FILE - the revision spec file in the appropriate language (specified during init0
8	—pull	Perform git pull before building
9	jobs purge <job name> [<branch name>]	Purge the build artifacts in conformance with the retention specifications If no branch name is specified, default branch
10	jobs show <jobname>	Complete log of all the builds
11	jobs enumerate —current —job repo|job	Provides the list of branches
	—job	Arg is a job. Otherwise arg is a job previously initialized
	—current	Current branch - use with job

Req ID	Command Line Example	
	jobs remove <jobname>	Remove any artifacts of a job previously initialized
	Jobs fetch <jobname> —set-default <branch>	clone a branch which is not the default branch Optionally set this branch as the default going forward for the job

Learning Objectives

- ☑ Manipulate, navigate directory structures in a platform independent way.
- ☑ Creating, storing configuration files in a standard format - toml
- ☑ Executing external commands and capture their output
- ☑ Exceptions as a means of informing error conditions
- ☑ Programmatically Interface to cli utilities like git

Examples

Initialize a repository

```
bin/jobs init git@gitlab.com:ada23/pwdgen.git
A blue on yellow line
Repository to setup git@gitlab.com:ada23/pwdgen.git
Repo dir is /Users/rajasrinivasan/jobspace/pwdgen
/Users/rajasrinivasan/jobspace/pwdgen is not a directory. Creating
/Users/rajasrinivasan/jobspace/pwdgen Is the repository root
Cloning git@gitlab.com:ada23/pwdgen.git /Users/rajasrinivasan/
jobspace/pwdgen Branch main
git@gitlab.com:ada23/pwdgen.git
Executing clone --branch main git@gitlab.com:ada23/pwdgen.git main
Cloning into 'main'...
```

After performing the initialization, the following directory structure with configuration files may be expected:

```
find $JOBSPACE/pwdgen -name "jobs.*"
/Users/rajasrinivasan/jobspace/pwdgen/jobs.toml
/Users/rajasrinivasan/jobspace/pwdgen/main/jobs.toml
```

Building the application

Once initialized, we can build:

```
bin/jobs build jotto build.sh
Build job=jotto branch:  script build.sh
Loading branch config from /Users/rajasrinivasan/jobspace/jotto/
master/jobs.toml
Next Build           1 Last built
Build directory /Users/rajasrinivasan/jobspace/jotto/master/1
Created dir /Users/rajasrinivasan/jobspace/jotto/master/1
```

As shown above, the utility does not know anything about the build steps. The application maintains a set of scripts for its build action. In the above, build.sh is a script that is executed.

Once the above step is performed, a log file is created storing the results of performing the action.

```
find $JOBSPACE/jotto -name "jobs.*"
/Users/rajasrinivasan/jobspace/jotto/jobs.toml
/Users/rajasrinivasan/jobspace/jotto/master/jobs.toml
/Users/rajasrinivasan/jobspace/jotto/master/1/jobs.log
```

Each build results in a separate directory based on its build number. A log is created in this directory. In this example, the log shows:

```
********jobs*********Created 2022-06-08 05:10:13.00*********
2022-06-08 05:10:13.00 ......> [I] Build Started
2022-06-08 05:10:13.00 ......> [I] Executing /Users/rajasrinivasan/
jobspace/jotto/master/scripts/build.sh
2022-06-08 05:10:18.00 ......> [I] /Users/rajasrinivasan/jobspace/
jotto/master/1
1
/Users/rajasrinivasan/jobspace/jotto/master
/Users/rajasrinivasan/jobspace/jotto/master/1
Note: Synchronizing workspace...
Nothing to update.

Setup
   [mkdir]        object directory for project Jotto
   [mkdir]        exec directory for project Jotto
Note: Building jotto/jotto.gpr...
Compile
   [Ada]          jotto.adb
   [Ada]          cli.adb
   [Ada]          words.adb
   [Ada]          words-dealer.adb
```

```
   [Ada]          words-player.adb
   [Ada]          revision.ads
Bind
   [gprbind]      jotto.bexch
   [Ada]          jotto.ali
Link
   [link]         jotto.adb
Build finished successfully in 2.46 seconds.
```

Of course a different script can perform something else altogether as shown in the following example:

```
% bin/jobs build jotto publish.sh
Build job=jotto branch:  script publish.sh
Loading branch config from /Users/rajasrinivasan/jobspace/jotto/
master/jobs.toml
Next Build           2 Last built 2022-06-08 09:10:13
Build directory /Users/rajasrinivasan/jobspace/jotto/master/2
Created dir /Users/rajasrinivasan/jobspace/jotto/master/2
rajasrinivasan@Rajas-MBP jobs % ls $JOBSPACE/jotto/master/2
jobs.log jotto        jotto.2.zip
rajasrinivasan@Rajas-MBP jobs % cat $JOBSPACE/jotto/master/2/jobs.log
*********jobs*********Created 2022-06-08 05:16:56.00*********
2022-06-08 05:16:56.00 ......> [I] Build Started
2022-06-08 05:16:56.00 ......> [I] Executing /Users/rajasrinivasan/
jobspace/jotto/master/scripts/publish.sh
2022-06-08 05:16:56.00 ......> [I]   adding: jotto (deflated 69%)
/Users/rajasrinivasan/jobspace/jotto/master/scripts/publish.sh: line
4: popd: directory stack empty
rajasrinivasan@Rajas-MBP jobs % unzip -l $JOBSPACE/jotto/master/2/
jotto.2.zip
Archive:  /Users/rajasrinivasan/jobspace/jotto/master/2/jotto.2.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
  1130008  06-08-2022 05:16   jotto
---------                     -------
  1130008                     1 file
```

Ada Implementation

git clone https://ToprLLC@bitbucket.org/ToprLLC/jobs.git